DOCUMENT RESUME

ED 182 788 CS 502 812

AUTHOR	Lipinski, Hubert; And Others
TITLE	Interactive Group Modeling: Part 1. Extending Group Communication through Computers.
INSTITUTION	Institute for the Future, Menlo Park, Calif.
SPONS AGENCY	National Science Foundation, Washington, D.C. Div. of Mathematical and Computer Science.
REPORT NO	R-44
PUB DATE	Aug 79
GRANT	MCS77-01424
NOTE	30p.; For related documents see CS 502 812-815
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	*Communication (Thought Transfer); *Communications; *Computer Programs; *Computers; Conferences; Futures (of Society); *Group Relations; Interaction; Networks; Research Projects; Telecommunication; *Teleconferencing

ABSTRACT

This is the first of four reports of the Interactive Group Modeling project--an undertaking designed to extend group communication through computers to support the more task-focused communication required by those building computer models. This report discusses the design and philosophy of the HUB system--a system that facilitates communication in four modes: computer conferences, shared visual spaces, program workspaces, and document workspaces. The various sections of the report provide a discussion of the needs for extending group communication and descriptions of the HUB system, design issues related to it, its implementation, and its testing. (FL)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED182788

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

INTERACTIVE GROUP MODELING:

PART 1. EXTENDING GROUP COMMUNICATION THROUGH COMPUTERS

Hubert Lipinski
Robert Plummer
Kathleen Spangler Vian

Institute for the Future
2740 Sand Hill Road
Menlo Park, California 94025
415-854-6322

August 1979

Report R-44

CS502 8/2

This report was prepared under Grant No. MCS77-01424, National Science Foundation, Division of Mathematical and Computer Science. The project is under the direction of Hubert Lipinski and Roy Amara. Other staff members who contributed to this report include Roberta Edwards, Barbara McNeal, Jeanne Muzzicato, Sara Spang, John Tydeman, and Carol Westberg.

This report is the first in the *Interactive Group Modeling* series. The other reports are:

Part 2. *An Interactive Monitor*
by Kathleen Spangler Vian, Hubert Lipinski, Robert Plummer, and Sara Spang, Report R-45

Part 3. *HUB and the Modeling Process*
by Kathleen Spangler Vian, Hubert Lipinski, Sara Spang, and John Tydeman, Report R-46

Part 4. *Some Preliminary Tests of the HUB System*
by Sara Spang, Barbara McNeal, and Kathleen Spangler Vian, Report R-47

The HUB User Guide, a manual for users of the HUB system

CONTENTS

	Page
ACKNOWLEDGEMENTS	v
FOREWORD	vii
EXTENDING GROUP COMMUNICATION: THE NEEDS	1
THE HUB SYSTEM: A DESCRIPTION	3
THE HUB SYSTEM: DESIGN ISSUES	7
THE HUB SYSTEM: IMPLEMENTATION	13
TESTING THE HUB SYSTEM: A PERSPECTIVE	23

Acknowledgements

This report marks the completion of the second year of our project on interactive group modeling, supported by a grant from the National Science Foundation. Although formally acknowledged only in this part of the report, the persons mentioned here have in fact contributed to all the phases of the modeling project. Within the Institute, we wish to thank Roy Amara for his help and advice over the course of the projects and Robert Johansen, whose ideas and suggestions have helped us focus our research directions. In addition, we wish to thank Carol Westberg for her editorial assistance, Roberta Edwards for her graphic art, and Jeanne Muzzicato, Valerie Dews, and Cynthia Anderson for their overall support.

We also are grateful to the members of our Advisory Board, Drs. Donald Austin, Saul Gass, Brian Mar, Julian Scher, and John Weyant, whose feedback, ideas, and suggestions proved very helpful both in the critique of our first-year report and in the direction of the final years of this research effort.

We owe a great deal of thanks to Dr. Fred Weingarten of the National Science Foundation for his capable monitoring of our projects and his suggestions for possible use of the research results.

Foreword

In the past six years, the Institute has been actively involved in assessing the social, economic, and managerial consequences of computer-based communication as provided by the PLANET system.* This area of research has run parallel to the Institute's main thrust of gathering and processing information about the future, constructing models that might describe the course of future events, and presenting the results to its clients in a form suitable for input to their own planning process. The objective of our current project on interactive group modeling is to combine the research in these two areas: we are extending group communication through computers to support the more task-focused communication required by people who are building computer models.

In the first year of this study, we have surveyed many of the reports that describe and critique computer modeling.** We have found that there are communication barriers among the model builders and the intended model users; there are also communication barriers among model builders themselves. Finally, there are communication problems associated with validation and documentation of models.

Based on this survey and discussions with groups doing modeling, we have developed the HUB system. This report looks at various aspects of the system--in Part 1, we discuss the philosophy and design of the system;

---

*See Jacques Vallee et al., *Group Communication through Computers, Volumes 3 and 4*, Institute for the Future, Reports R-35 and R-40, 1975 and 1977; and Robert Johansen et al., *Group Communication through Computers, Volume 5*, Institute for the Future, Report R-41, 1977.

**Hubert Lipinski, Roy Amara, and Kathleen Spangler, *Communication Needs in Computer Modeling*, published in conference proceedings, *1978 Winter Simulation Conference*, vol. 1, December 1978.

Part 2 introduces the concept of interactive monitoring of group communication and explores the uses and impacts of the HUB monitor; Part 3 identifies potential roles of HUB in the modeling process and offers a preliminary taxonomy for mapping its uses; and Part 4 assesses the communication process in our structured tests of the HUB system and proposes a future research agenda.

Extending Group Communication: The Needs

Most conventional computer applications support a single user in very structured tasks, such as creating and running programs, creating and interrogating data bases, and computer-aided design. In contrast, teleconferencing systems support unstructured communication among a group of people through the exchange of messages. If that group of people were jointly working on a model, their tasks could be facilitated by some of the structuring capabilities of the computer. The question is, which ones?

In moving from computer-based conferencing toward more task-focused communication within the HUB system, we were faced with a number of possible alternatives. They dealt with the choices of the specific task focuses for expanded communication and with the organization of the total system. While the choice of system organization could be made relatively independently, the selection of specific task focuses was heavily influenced by our research objective of trying to improve communication in the modeling process. The task focuses, in turn, would influence the capabilities our system would need to provide.

Some characteristics of existing computer conferencing systems are well suited to supporting communication among modelers. PLANET, for example, is a very simple system that allows a group of people to work together. The members of the group may work long distances from each other and may have widely varying skills. The transcript of PLANET conferences provides automatic documentation of the group's interactions.

But a system well matched to modelers' needs could require new capabilities, too. Such a system would be more task-focused; it would provide more flexibility and more safeguards in documentation; and it would give the group access to more structured computer resources. The need for certain resources is obvious: modelers often use flow charts and other visual

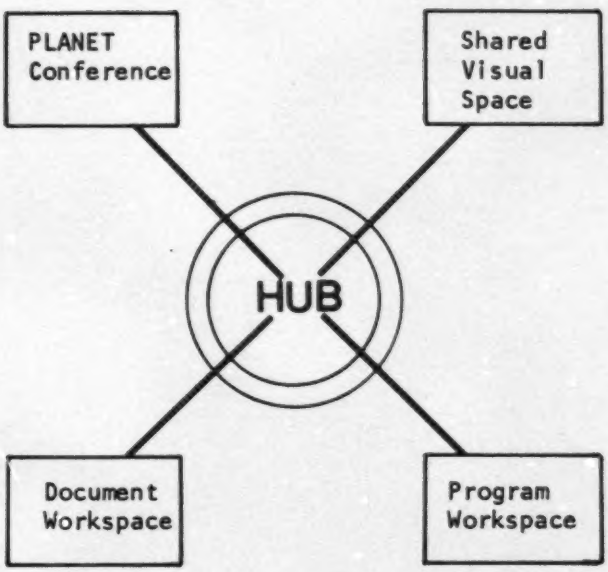
representations requiring graphics capabilities. They must prepare, review, and edit formal documents. They need special tools that will help in defining problems through determining important variables and estimating their interrelationships. They need to run existing packaged programs and to create and run programs of their own.

These are the needs that have guided our development of the HUB system; we expect to become aware of others as we exercise the system in field tests over the next two years. In addition, we believe that the expanded communication facilities we have developed have much wider applicability than the modeling perspective on which we focus.

The HUB System: A Description

The HUB system facilitates communication in four modes: PLANET conferences, shared visual spaces, program workspaces, and document workspaces, as shown in Figure 1.

FIGURE 1
THE HUB SYSTEM



These four task-specific communication facilities are organized as separate modules in the HUB system, which is named for its central component, the "HUB." The HUB module itself is an intelligent switcher that makes it easy for users to access and move among the subsystems attached to it. In

fact, we perceive the HUB system as a prototype for the management system of the future--a system in which a user can easily interact with a large number of computer-based resources. Once the user runs the HUB program, he or she has access to the following types of communication:

THE PLANET CONFERENCE

A primary need of a modeling group, especially at the beginning of its efforts, is simply to discuss the project, its goals, the approach to be taken, the underlying assumptions, and known variables. Accordingly, we have included under HUB the PLANET computer conferencing system developed at the Institute. PLANET provides this kind of open-ended communication. It allows the modeling team to "get together" without the time and travel constraints of face-to-face meetings; it also provides a complete transcript of the discussion. (PLANET as a communications medium has been extensively discussed in the Institute's publications.)

THE SHARED VISUAL SPACE

As the form of the model begins to emerge from initial discussions, a block diagram or flow chart can be useful in representing it. Graphical representation of the output from the model can also help the modeling team. Just as PLANET is oriented toward the exchange of text, the shared visual space is a subsystem oriented toward graphic communication. Using it, modelers will be able to create, view, exchange, and modify a wide variety of graphic images, including block diagrams, flow charts, bar charts, graphs, and, to some extent, sketches. Comments can be attached to the various parts of images, providing "on-the-spot" explanations of particular items or of changes that have been made.

We hope to implement the shared visual space by early 1980. When complete, this software will support a range of terminals, from those with sophisticated graphics capabilities to simple paper-printing teletypes.

THE PROGRAM WORKSPACE

Building a computer model includes the use of existing computer programs as well as the development of new programs. The program workspace is designed to aid in these two activities.

When a user enters the workspace and begins a program run, others may join him or her in a kind of "looking-over-the-shoulder" mode. While any number of onlookers can join and participate in the discussion, only one user effects changes in the program. All users will see the entire transcript of the program run on their own terminals; they can also exchange comments about the run as it proceeds. For example, if the program is a piece of the model, they might debate the values to be supplied for the model's parameters. The transcript of these "synchronous" runs (including comments) can be stored for later users who were not present.

This capability can facilitate joint development of programs, but equally important will be the ability to use a variety of existing computer-based premodeling tools. The early stages of the modeling process often call for a preliminary definition of the relationships among key variables. Simulation tools such as KSIM* and ISM** have been developed for just this purpose. In the program workspace, groups can work with such tools, trying out different assumptions before they are locked into a particular model structure.

THE DOCUMENT WORKSPACE

It's difficult to overstate the importance of good documentation to a modeling project. If the model's goals and assumptions are to be made clear, if users are going to be able to maintain the model, and if it is to serve the needs of the model users, then documentation is essential.

---

*J. Kane, "A Primer for a New Cross-Impact Language - KSIM," *Technological Forecasting and Social Change*, vol. 4, 1972, pp. 129-42; and H. Lipinski and J. Tydeman, "Cross Impact Analysis--Extended KSIM," *Futures*, vol. 11, no. 2, April 1979.

**R. H. Watson, "Interpretative Structural Modeling - A Useful Tool for Technology Assessment?," *Technological Forecasting and Social Change*, vol. 11, 1978, pp. 165-85.

The HUB system as described so far aids documentation in several ways. The discussions, explanations, and transcripts of program runs are all saved and can be reviewed later in a structured manner. The document workspace focuses on another aspect of the problem: the preparation of the modeling project's "formal" documents. In this subsystem, documents can be prepared, reviewed, and edited using a simple, line-oriented text editor. Furthermore, users can hold a PLANET-like discussion about the documents in order to make comments or criticisms.

One important note: our discussion of the other subsystems has been roughly "chronological." Most modeling projects will proceed from general discussion, to graphic representation, to a computer program. The document workspace breaks this progression, since an effective modeling team will produce documents throughout the modeling effort.

* * *

These four kinds of services provided by HUB are designed to facilitate communication within a group working on a joint task. Given this basic framework, we now turn to some of the more specific design choices we have faced.

The HUB System: Design Issues

The extension of computer-based group communication to task-focused objectives raises a large number of design issues. These issues range from the choice of specific task focuses to the characteristics of the modules that support various types of communication. Within that range, the issues concern both human-to-human and human-to-machine interaction.

Our choices of specific task focuses are a direct result of our assessment of modeling needs, solutions various researchers have proposed, and our own perceptions resulting from work in both the communications and modeling areas. To select effective extensions to unstructured group communication, we have examined the technical capabilities that were required to support the activities that modelers undertake. On the basis of this analysis, we have designed the HUB system to facilitate the most important capabilities. Each module is intended to perform one primary task, and the modeler can carry out various activities by integrating the tasks.

In addition to these basic choices of focus, we faced the problem of designing the specific modules. Our previous research in unstructured group communication through computers has revealed a number of general communication characteristics that are critical to preserve in moving toward more structured communication. First, the structure should allow for synchronous/asynchronous communication; the communication modules must function equally well when many persons are "present" as when there is only one. Second, the system must preserve a record of the interaction, not only to update people who were not present, but also to allow for search and retrieval of specific information. And third, any structure that facilitates computer-based group communication should assume the participants may be geographically dispersed. Each of these characteristics will impose certain requirements on a system designed for task-focused group communication.

SYNCHRONOUS/ASYNCHRONOUS COMMUNICATION

Whether or not real-time communication is planned, traditional eight-hour working days usually ensure that some fraction of electronic communication occurs when the sender and receiver are "present" at the same time. The real-time capability seems to be especially important in group communication processes in which people hold a meeting in order to reach closure on some particular problem. Starting from the premise that all participants will be present at the same time, we have designed the various communication modules to support such synchronous communication--which should produce a smoothly functioning system even when people are not present at the same time. Synchronous communication has implications for the user as well as the software.

At a user level, it involves facilitating interaction among many people at the same time. Procedures that are perfectly adequate for one person often prove cumbersome and aggravating when many are present simultaneously. PLANET deals with this difficulty by automatically displaying status changes (such as when a participant enters or leaves a discussion or starts typing), printing new messages to the user's terminal automatically, requiring no special procedures to begin entering public messages, and only sending a message after the user ends it with two consecutive carriage returns. In our extension to other communication tasks, we have attempted to continue these approaches not only to facilitate the interaction but also to provide a unifying theme in the modules.

At the software level, our work with PLANET has demonstrated that synchroneity requires file protocols that allow many users to read and write data simultaneously. Consequently, the conventional file protocols must be bypassed, the files that contain the record of the interaction must be random-access files, the information in these files must be directoried, and all critical operations on them must be done in a manner that avoids conflicts. These protocols, which are most easily carried out by coding the file handling subroutines in assembly language, again underlie the software in the new communication modules.

PRESERVING A RECORD

A written record is essential to the functioning of a computer-based communication system. It allows people to participate at their own convenience (that is, asynchronously) as well as to search for specific information that has been entered into the communication process. Providing such a permanent record is quite straightforward in the PLANET communication mode. The system stores messages sequentially as they are entered in an activity file and maintains a set of message counters to keep track of who has seen what. In the other existing modules, however, preserving a record is not such a simple procedure.

The focus of the program workspace is the running of a computer program. However, a stored transcript of such a program run is usually not available since program output is normally directed to and from a user's terminal. Thus, to preserve a record of a program run, we found it necessary to send the program output first to a file and then pass it on to the user. Although this procedure allows the program transcript to be recorded, it does not permit others to view the interaction or comment on it. This complication results since under the operating system we are using, the file can only be shared between the user and the program he or she is running, not with others.

To provide for a written record that encompasses not only the interaction between a user and a program, but also the comments of other participants, we simulate a PLANET-like discussion. The program run transcript is treated as the input of a participant and is automatically entered into a discussion activity. Likewise, individuals' comments can be entered and the same procedures that generate the record of a PLANET discussion can be invoked to preserve the run transcript.

In the document workspace, the issue is not so much how to keep a record as what record to keep. One could keep every version of the document but the transcripts would soon become unwieldy, from both a storage and a user standpoint. The design question we asked here is: Just what does the user need to be kept up to date? That information should become the record.

From our perspective, the changes that are made to the document are the most important focus of work in the document workspace. Moreover, since

people not only wish to modify a document, but also want to talk about it, we need to add to the record of changes the comments of various participants. Both the changes and comments are preserved as entries in a PLANET-like discussion, with the changes being automatically recorded when a user has finished editing the document. In addition, the current version of the document is available whenever a participant finds that there have been too many changes for him or her to absorb.

In all three cases, the communication module keeps track of what information each user has seen and displays only the new information. However, the entire record is available for review using a wide variety of retrieval specifications. Thus, the preservation of the communication record may well serve as an automatic documentation device--available in *all* submodules--that can greatly assist a group involved in a modeling effort.

GEOGRAPHIC DISPERSION

To make PLANET available to participants who are geographically dispersed, we originally installed the program on computers that could be reached through national and international computer networks. We followed a similar rationale with the HUB system. Thus, HUB may be reached either through the commercial TELENET network or through the ARPANET.

However, in the program workspace, participants include not only people, but programs, which can be located at other computers. Thus, it was necessary to include a mechanism not only for running programs on the computer where the HUB system resides, but also for running programs on other computers.

Since many of the user groups with which we hope to test the HUB system have access to the ARPANET, we included a facility through which users can run programs on any computer available through the ARPANET. Even though HUB resides on a computer in Boston, a user group could easily work with a model located on a computer in, say, Los Angeles. The long-distance communications between HUB and the program in L.A. would be barely perceptible to the user.

For users who do not have access to the ARPANET and wish to use a program residing on a different computer from the one on which HUB resides, it may be possible to use a terminal with internal storage to record the program run and then enter it into the program workspace. For example, this turned out to be a fairly simple process with a Texas Instruments Model 765 bubble-memory terminal. However, using such a terminal precludes synchronous interaction focused on the program run. Since many potential users of a HUB-like system might not have access to the ARPANET or use computers not on the ARPANET, we think it is vitally important to design a standard to permit user-initiated computer-to-computer communication. This is made all the more urgent since we are rapidly moving into the world of distributed and local data processing.

EASE OF USE OF COMPUTER RESOURCES

The extension of group communication to structured tasks requires the use of other computer-based resources, which must be integrated into the HUB system. Our approach to the problem has been to make the integration as unobtrusive as possible.

In the program workspace, any participant can run programs simply by asking for the RUN (program) service and typing the name of the program. The HUB system then carries out a set of instructions needed to start the program; this set is entered when the program is added to the workspace by the person who is familiar with it. The procedure for running a program on the local computer is identical to that for a remotely located computer. Once a program has been added, it is available to anyone in the workspace without special instructions. Thus, the HUB system tends to shift the focus from where the program resource is and what machine it runs on to what the program can do.

In the document workspace, the HUB system uses the NETED editor commands to modify the document. The user makes changes in the document without awareness that the editor is being called up, that the document is being entered into the editor's workspace, and that control is relinquished to the

editor. When the user is finished, the new changes are entered automatically into the workspace proceedings. Again in this case, access to the resource is made as simple as possible.

STRUCTURAL ORGANIZATION OF THE COMMUNICATION PROCESS

The structural organization of a PLANET conference is quite simple. The messages are units in a sequential flow of information. In extending the characteristics of other task-related communication, we initially adopted a similar premise, but soon found it unworkable. Each module required a different organizational structure to best meet its communication objectives.

In the program workspace, the main units of information are the program run transcripts. Each transcript includes the record of the program interaction, with users' comments interspersed or appended to it. The rationale for this choice is that in this mode, the users are interested in the parameter choices and corresponding outputs of each program run and hence will focus their comments on them.

In the document workspace, the changes to the document are recorded sequentially. The comments, however, are attached to various changes. The basic unit of communication, then, is the change and all its attendant comments. In addition, the changes and comments are not presented in the order they were entered; rather, the new or reviewed information is sorted by page and line number. This approach puts the information in a context similar to that of the document it describes.

* * *

The needs of a group working together on a task have clearly influenced our design of the HUB system. We expect that user groups who try out HUB over the next two years will give us feedback as to how the system does and does not meet their needs. A critical part of that match will be how smoothly HUB actually runs.

The HUB System: Implementation

The design decisions we have discussed require certain technical capabilities in the HUB system. The design process is not one-directional, however, as what is possible technically certainly has influenced our design ideas. To minimize the constraints, we have implemented HUB on a computer operating system with as few restrictions as possible. Thus, we are using the PDP-10 computer at Bolt Beranek and Newman in Boston, which runs the TENEX operating system.*

This system provides the capability: (1) to allow parallel running processes through a fork structure; (2) to map pages between forks and files; (3) to reset the primary input and output device from the default of the user's terminal to a file; and (4) to share files among processes. While our emphasis on these types of capabilities may be the result of our specific choice of operating system, we believe that they are indicative of what is necessary to support any task-focused group communication through computers. The need for these capabilities will become apparent as we discuss the technical aspects of the individual modules in the HUB system.

THE HUB MODULE

The user interacts with the central HUB module, an intelligent switcher that routes the user to the appropriate communication module. It signs the user into each of the individual modules and facilitates movement between them. In fact, its objective is to minimize any sense of discontinuity so the user can concentrate on the tasks involved in the modeling/communication process.

---

*Our experience to date suggests that we have sacrificed reliability of operation in order to achieve the freedom we feel necessary. Ideally we would like flexibility *and* reliability, but the real world doesn't seem to offer both.

In order to sign the user into the submodules, the HUB switcher asks for the user's name and password. With this information, the module scans the activity files of each of the submodules to determine the activities in which the user is registered and any new information in those activities. It then prints a list of the activities, identifying whether each is a PLANET conference, document workspace, or program workspace.

Once the user indicates the activity he or she wishes to join, the HUB module creates a lower fork, places the desired submodule into it, copies the sign-in and other necessary information into the appropriate storage areas in the submodule, and then starts the activity. Once the HUB module has relinquished control, it goes dormant until the submodule terminates. Then, depending on what the user desires, it enables him or her to join another activity or ends the user's participation in the HUB system.

THE PLANET SUBMODULE

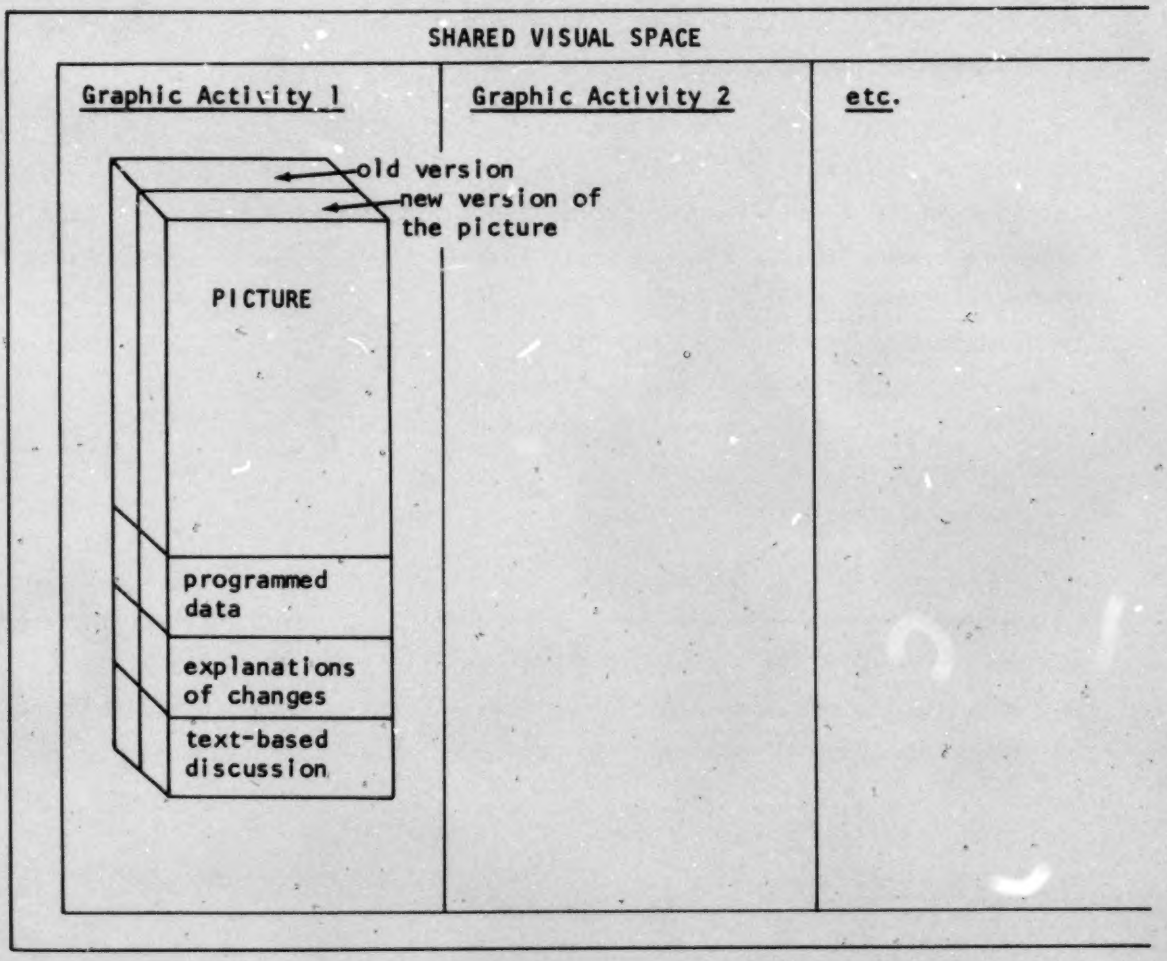
This submodule is the same program that has supported our earlier work in unstructured group communication. The technical details of its implementation have been described in earlier publications and form the basis for our extension to other task-focused communication processes. Many of these considerations will be discussed as we consider the other submodules.

THE GRAPHICS SUBMODULE

Although the graphics submodule has not yet been implemented (as separate funding is being sought), we sketch here our preliminary ideas on the technical aspects needed to support this form of communication. Our approach to developing a computer-based graphic communication system is to design a shared visual space, which will be an organized set of interactive graphic activities. The focus of each activity will be an evolving picture. By entering a graphic activity, users can observe or participate in the evolution of the picture; they can view the latest version (or earlier versions), define new parts of the picture, modify the existing picture, or comment on it in text messages. This concept is illustrated in Figure 2.

FIGURE 2

CONCEPT OF THE SHARED VISUAL SPACE



When users enter a graphic activity, they begin to interact, through a terminal keyboard, with a system that we call the graphics organizer. They communicate with this graphics organizer using a picture description language. Each picture consists of a set of graphics primitives written and stored as text. Using this description, the graphics organizer produces the picture on the user's terminal screen or paper printout.

In addition to producing pictures and storing their descriptions, the graphics organizer performs a number of other functions. It prompts the user during the picture description process; for example, it will ask for values for the parameters of primitives, thus relieving the user of the responsibility of remembering the order of parameters or even what they are. In addition, it manages text messages and may even prompt the user for such messages in order to document changes to the picture. In short, it organizes all of the communication within a graphic activity.

Each graphic activity will reside in a separate computer file, accessible by the graphics organizer. The file will contain all the picture descriptions and messages associated with the graphic activity. Because several users might wish to work in the graphic activity at the same time, the file will be open to everyone. This means that a mechanism to prevent users from interfering with each other must be implemented in the graphics organizer program and in the structure of the activity file. This approach provides the flexibility we believe we need when the ultimate protocols for using the system are not well understood.

The graphic activity will be a random-access file that will be organized in a structured fashion. The graphics organizer program will access the picture descriptions and messages through information contained in various directories. There will be two main directories, one for the picture descriptions and one for text messages. In addition to providing the location of the appropriate text, both directories will contain information on the name of the author and the date and time of entry. The picture description directory will provide the previous picture version that was modified, and the message directory will key the part of the picture to which the message refers. Secondary directories will link the picture descriptions to the messages that reference them.

Interference of one user with another will be prevented by the creation of new versions of picture descriptions. When a user enters an activity, the current version of the picture (or any other version desired) will be copied into a buffer area. If the user changes the picture, then this buffer will be copied into the picture description file when the user leaves the activity, and an appropriate directory entry will be made.

All critical writing operations such as updating the number of entries in a directory will be performed in a conflict-free manner. Because of this necessity and the nature of the activity file structure, the subroutines that read and write information from the activity file will be coded in assembly language. The remainder of the program will be written in a higher-level language.

THE PROGRAM WORKSPACE SUBMODULE

The program workspace is by far the most complicated module in the HUB system because it supports communication not only among a group of persons but also with computer resources that may be remotely based. It combines these communication channels so that the user perceives only one interaction. This principal function of the program workspace is supported through the RUN service.

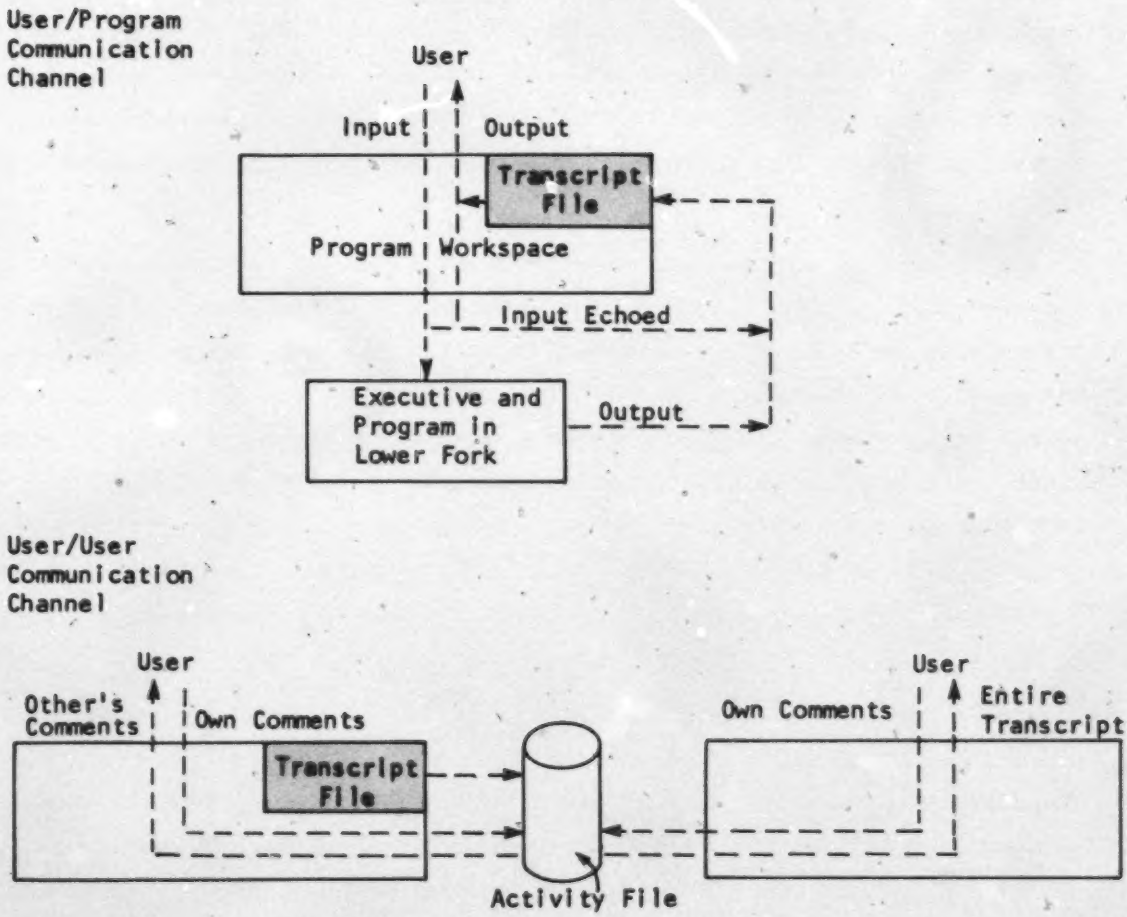
The basic philosophy of the RUN service is to permit a user to run a program as simply as possible and allow others to view and comment on the interaction. To make running a program simple even for the unfamiliar user, the participant adding the program to the workspace is asked to enter into HUB the necessary operating system instructions for starting the program run. Thus, a user simply types the program name in response to the RUN (program) prompt, and HUB then automatically carries out the preset instructions.

To support communication focused on the running of a computer program requires two communication channels: one between the user and the computer resource and another among all participants. In the case of a program residing on the same computer as HUB, the procedure for creating this communication channel between the user and the program is illustrated in Figure 3. (The procedure is actually simpler in the case of a remotely based resource.) HUB creates a lower fork in which the computer resource will be run, creates a file in which to store the transcript of the interaction, and changes the destination of the lower fork's output from the default setting of the user's terminal to the file. Thus, output from the lower fork will be passed to the file. Similarly, since the input and output devices are not identical

for the lower fork (input is from the terminal, output is to the file), input is echoed not only to the user's terminal but also to the output device and thus into the file. The interaction is now recorded, but since the program output is directed to the file, additional steps are needed so that the user sees what the program prints.

FIGURE 3

RUNNING A PROGRAM ON A LOCAL COMPUTER
AND COMMENTING ON THE RUN



In order to have the user see the output of the interaction, HUB maps pages of the record file into its core in the upper fork. HUB constantly watches the file until it increases in size and then reads the additional characters from its core area and sends them to the user's terminal. (It has to recognize the user's input, however, since that has already been echoed as it was typed.) Thus, the user sees the interaction, which is being recorded as well.

To permit others to see the interaction, HUB creates a temporary PLANET-type activity file and automatically enters the transcript of the interaction into it. As new material appears in the file, HUB formats it into a message and enters it into the activity file. Each message is at least one line long and may only contain an integer number of lines. This preserves system efficiency and allows users' comments to be interspersed in the transcript record in a fairly orderly fashion. Comments are written directly into the temporary file as they are finished and appear in the transcript at that point.

For users who are only watching, the commenting is straightforward. They view the emerging transcript and make comments. For the user running the program, however, the situation is slightly more complicated. That person may "talk" to only one process at any one time. Thus, he or she must switch from one communication cha nel to the other in order to make a comment about the interaction. To do this, the user types a CONTROL Z, which has been enabled as an interrupt. HUB freezes the lower fork, suspending the program, and then assumes that any further inputs are in fact comments. Once finished, the user types another CONTROL Z. HUB then unfreezes the program and passes all input to the program again.

When the program run is completed, the user is offered the option of storing the entire transcript. If so instructed, HUB copies the transcript (both the text and the directory entries that reference the text) into the permanent workspace activity file. The transcript is then available for review and commenting.

THE DOCUMENT WORKSPACE SUBMODULE

The objective of the document workspace is to allow a group of users to create a document and talk about it. Technically it supports two channels of communication: one between a user and the editing program and another among the participants in the workspace. However, in designing the workspace, we were faced with the problem of what to do if several users wish to edit the document simultaneously.

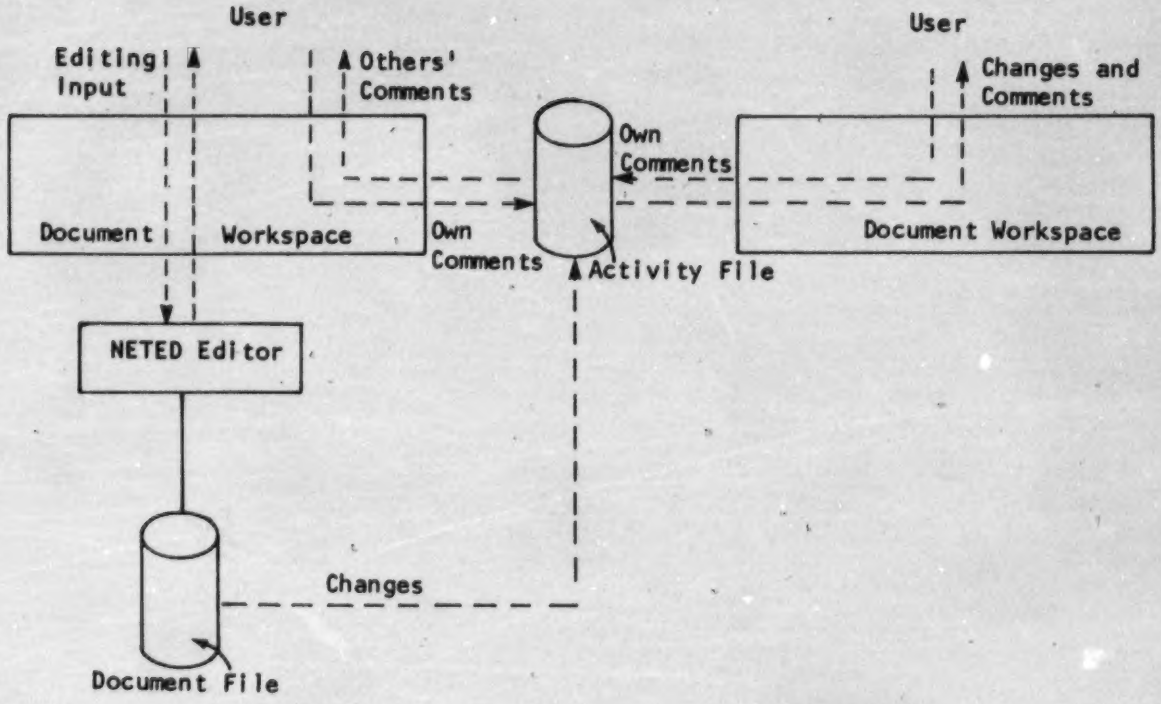
The usual solution takes one of two forms: allow only one user to edit at a time or allow all to edit, creating many new versions of the document. Because the document workspace provides a channel of communication among the participants, we have chosen an intermediate alternative. Only one user can edit at a time, but other participants can send the user messages (and receive messages as well) during the editing.

The editing changes that a user makes are currently automatically recorded in the workspace transcript as soon as the user finishes editing. Our initial hope was to record the changes in a more continuous manner while the user was editing. This would have made our solution to the simultaneous editing problem far more appealing. However, while technically straightforward, this procedure is cumbersome both from a software and a user standpoint. We have decided to wait for some hard evidence from the evaluation as to what is most useful.

To allow a user to edit the document, HUB first creates a lower fork, starts the NETED editing program in the fork, and supplies NETED with the name of the file in which the document is stored. It then directs input to the editing program in the lower fork. At the same time, in the upper fork HUB has opened the communication channel to other participants who might be present or join later. This communication is carried out through the workspace activity file, which is organized quite similarly to a PLANET conference file. While editing, users may receive comments from other participants and, as in the program workspace, may suspend the editor to make comments themselves. The entire process is illustrated in Figure 4.

FIGURE 4

EDITING A DOCUMENT AND
COMMENTING ON THE CHANGES



As already mentioned, HUB automatically enters the changes made in the document into the workspace activity. It does so by comparing the old and new versions of the file line by line. Currently it records changes as insertions, deletions, and replacements of text, and these changes form the principal entries in the document workspace record. When participants make comments, HUB asks which changes the comments refer to and links them to the appropriate changes.

Testing the HUB System: A Perspective

There is no real methodology for extending group communication from the unstructured base from which we started. We adopted a number of particular approaches and some very specific implementations based on our own and others' perceptions. We need to take a "try and test" approach. In particular, we need to explore the balance among simplicity, power, and flexibility.

The technical choices that we have made have spoken directly to simplicity and power. We've designed simple-to-run programs. We've created the power for groups to run programs together, and the power to use the operating system directly. However, some decisions have created inflexibilities in communication channels.

One limitation that has been mentioned with the program workspace is that remotely located programs must reside on computers that are on the ARPANET. For locally residing resources, the compilers used to create the object code module can cause problems. For example, the new DEC FORTRAN compiler is compatible with the program workspace, but the older F40 is not. The BASIC compiler, which is similar to F40 in implementation, also does not work. A user can run a LISP program, but while the interaction appears perfectly normal to others, characters sometimes double print on the user's own terminal. The difficulty stems from whether the program reads and responds a line or a character at a time. Fortunately, these problems do not occur with remotely based resources.

Such technical restrictions, while not large in themselves, exacerbate the constraints introduced by our design ideas. The real question we must answer is whether the main thrust of our extensions to computer-based communication provides the communication flexibility required to be useful in a variety of modeling efforts. Such flexibility is not only required at the

technical level of the communication channel (which will never be able to perform the communications function exactly as each user would like), but is also crucial at the more social levels of task and institutional setting. Does HUB fit within a variety of conceptions about what the modeling (and other) tasks are and how they are to be accomplished? Does it fit into the procedures that have already been developed by various people and institutions to deal with their communication needs? It is to answer these types of questions that we are assessing the communications flexibility of the HUB system in the final two years of the project.